Λειτουργικά Συστήματα
2018 - 2019
1η Εργαστηριακή Άσκηση
Ημερομηνία Παράδοσης: 7/1/2019

Η παράδοση της άσκησης θα πραγματοποιείται με αποστολή μηνύματος ηλεκτρονικού ταχυδρομείου **ΚΑΙ ΣΤΙΣ ΤΡΕΙΣ** ακόλουθες διευθύνσεις **με ένα μήνυμα** (με τρεις παραλήπτες και όχι τρία διακριτά μηνύματα):

sioutas@ceid.upatras.gr, makri@ceid.upatras.gr, aristeid@ceid.upatras.gr

Μπορείτε να συντάξετε την αναφορά σας για τα θέματα 2, 3 και 4 σε όποια μορφή κειμένου επιθυμείτε (word, pdf, κ.λπ.). Στο ηλεκτρονικό μήνυμα που θα αποστείλετε θα έχετε συμπιεσμένα το αρχείο της αναφοράς σας και το αρχείο του shell script (θέμα 1).

Μέρος 1 [35 μονάδες]

Γράψτε ένα πρόγραμμα σε BASH shell για τη διαχείριση ενός αρχείου χρηστών καταγραφής γεγονότων (log files) για την πλοήγηση σε κοινωνικά δίκτυα στο διαδίκτυο. Τα αρχείο, με όνομα **event.dat** (βρίσκεται τοποθετημένο στο eclass), που θα επεξεργάζεστε περιέχει στοιχεία σύμφωνα με την ακόλουθη γραμμογράφηση και μορφοποίηση:

```
id|lastName|firstName|gender|birthday|joinDate|IP|browserUsed|socialmedia
```

Γραμμές που ξεκινούν με # θεωρούνται σχόλια και πρέπει να αγνοούνται. Το πρόγραμμα σας θα πρέπει να ονομάζεται **tool<AM>.sh**, όπου <AM> προκύπτει από τα τρία τελευταία ψηφία του αριθμού μητρώου κάθε μέλους της ομάδας των δύο ατόμων, δηλαδή εάν το ένα μέλος έχει ΑΜ 1234567 και το άλλο μέλος έχει ΑΜ 1987654 τότε το όνομα του αρχείου θα είναι tool567654.sh

Το πρόγραμμα θα υποστηρίζει τις παρακάτω λειτουργίες:

1. [Μονάδες 5] Όταν το εκτελείτε χωρίς καμία παράμετρο, θα εμφανίζει τον ΑΜ κάθε μέλους της ομάδας σας, χωρισμένα με παύλα (-) και χωρίς ΚΑΝΕΝΑΝ άλλο χαρακτήρα (ούτε κενά). Παράδειγμα:

   ```
   1234567-1987654
   ```

   Η εκτέλεση της εντολής:

   `./tool.sh -f <file>, όπου <file> είναι το όνομα του αρχείου γεγονότων (π.χ., event.dat),`

   θα εμφανίζει όλα τα περιεχόμενα του αρχείου, χωρίς τις γραμμές-σχόλια που πρέπει να έχουν αγνοηθεί!

2. [Μονάδες 5] Η εκτέλεση της:

   `./tool.sh -f <file> -id <id>`

   θα εμφανίζει το όνομα, επώνυμο, και ημερομηνία γέννησης του χρήστη με το συγκεκριμένο id, χωρισμένα με ένα μόνο κενό. Για παράδειγμα:

   `$ ./tool.sh -f persons.dat -id 10995116283604`

   Θα εμφανίζει:

   `Gilels Viktor 1982-04-18`

3. [Μονάδες 8] Η εκτέλεση της

   `./tool.sh --firstnames -f <file>`

   θα εμφανίζει όλα τα *διακριτά* πρώτα ονόματα (πεδίο firstname) που περιέχονται στο αρχείο, σε αλφαβητική σειρά, ένα ανά γραμμή και η εκτέλεση της:

   `./tool.sh --lastnames -f <file>`

θα εμφανίζει όλα τα *διακριτά* επώνυμα (πεδίο lastname) που περιέχονται στο αρχείο, σε αλφαβητική σειρά, ένα ανά γραμμή.

4. [Μονάδες 7] Η εκτέλεση της εντολής:

   `./tool.sh --born-since <dateA> --born-until <dateB> -f <file>`

   Θα εμφανίζει μόνο τις γραμμές που αντιστοιχούν σε χρήστες που έχουν γεννηθεί από την ημερομηνία dateA μέχρι την ημερομηνία dateB. Μπορεί να δοθεί και το ένα από τα δύο, π.χ., για να εμφανίσει όλους του χρήστες γεννημένους από μια μέρα κι έπειτα, ή γεννημένους μέχρι κάποια μέρα. Για κάθε χρήστη που πληροί τα κριτήρια, να εμφανίζεται αυτούσια η γραμμή όπως ακριβώς ήταν στο αρχείο. Προφανώς δεν θα περιλαμβάνονται σχόλια.

5. [Μονάδες 10] Η εκτέλεση της:

   `./tool.sh --socialmedia -f <file>`

   θα εμφανίζει όλα τα μέσα κοινωνικής δικτύωσης (social media) που χρησιμοποιήθηκαν από τους χρήστες, σε αλφαβητική σειρά, και δίπλα από την αναφορά του ονόματος του κοινωνικού δικτύου θα αναγράφεται ο αριθμός των χρηστών που τον χρησιμοποίησαν (με ακριβώς ένα κενό για χώρισμα). Για παράδειγμα:

   `Facebook 527`

   `Twitter 696`

   Η εντολή:

   `./tool.sh -f <file> --edit <id> <column> <value>`

   θα μεταβάλει το αρχείο. Συγκεκριμένα, για τον χρήστη με κωδικό <id>, θα αντικαθιστά τη στήλη <column> με την τιμή <value>. Αν δεν υπάρχει κανένας χρήστης με αυτό το id, ή η στήλη δεν είναι μεταξύ των αποδεκτών στηλών 2 έως 8 (η στήλη 1 που αντιστοιχεί στο ίδιο το id δεν επιτρέπεται να αλλάξει), η εντολή αυτή δεν θα μεταβάλει τίποτα. Πέρα από τη ζητούμενη αλλαγή, δεν θα πρέπει να μεταβάλεται τίποτα άλλο, δηλαδή πρέπει να διατηρείται η αρχική ταξινόμηση των εγγραφών συμπεριλαμβανομένων των σχολίων.

   Σε όλα τα παραπάνω, οι παράμετροι πρέπει να μπορούν να δίνονται με οποιαδήποτε σειρά. Για παράδειγμα τα παρακάτω είναι ταυτόσημα:

   `./tool.sh --born-since <A> --born-until <B> -f <F>`

   `./tool.sh --born-until <B> -f <F> --born-since <A>`

   `./tool.sh –f <F> --born-until <B> --born-since <A>`

   `./tool.sh –f <F> --born-since <A> --born-until <B>`

Προσοχή

- Η έξοδος που παράγει να είναι αυστηρά αυτή που ζητείται.
- Τεκμηρίωση (documentation) **ΜΗ** δώσετε ξεχωριστά, αλλά να έχετε επαρκή σχόλια μέσα στον κώδικα.

Μέρος 2 [30 μονάδες]

Ερώτημα Α [10]

Δίνεται η ακόλουθη αριθμητική έκφραση:

```
Y = G + X * (C + D) / ((E - F) * (H + I))
```

Η έκφραση υπολογίζεται από τον παρακάτω κώδικα που εκτελεί ακολουθιακά τις εξής 7 εντολές (I1; I2; ….; I7;):

```
I1: Y11 = C + D;
I2: Y12 = E - F;
I3: Y13 = H + I;
I4: Y21 = X * Y11;
```

```
I5: Y22 = Y12 * Y13;
I6: Y31 = Y21 / Y22;
I7: Y = G + Y31;
```

Στον παραπάνω κώδικα χρησιμοποιούνται, εκτός από τις μεταβλητές Y, X, C, D, E, F, G, H και I που παρουσιάζονται στην αριθμητική έκφραση, και οι μεταβλητές Y11, Y12, Y13, Y21, Y22 και Y31 για τον υπολογισμό των ενδιάμεσων αποτελεσμάτων της αριθμητικής έκφρασης.

[i] Να σχεδιάσετε ένα γράφο προτεραιοτήτων (precedence graph) που να μοντελοποιεί (με τη μέγιστη δυνατή παραλληλία) τη σειρά εκτέλεσης των ανωτέρω (I1-I7) εντολών. **[2]**

[ii] Με χρήση των εντολών begin … end και cobegin … coend **(χωρίς να χρησιμοποιήσετε σημαφόρους καθώς και εντολές P (ή wait ή down) / V (ή signal ή up))** να δώσετε ένα κώδικα που να υπολογίζει (με τη μέγιστη δυνατή παραλληλία) την ανωτέρω έκφραση. **[4]**

[iii] Να δώσετε ένα «παράλληλο» κώδικα, χρησιμοποιώντας την εντολή cobegin … coend καθώς και εντολές P (ή wait ή down) / V (ή signal ή up) σε σημαφόρους που θα ορίσετε, το οποίο να υπολογίζει την ανωτέρω έκφραση. **Μην παραλείψετε να αρχικοποιήσετε τους σημαφόρους που θα χρησιμοποιήσετε**. **[4]**

Ερώτημα Β [10]

Έστω οι παρακάτω δύο διεργασίες Δ1 και Δ2 που εκτελούνται «παράλληλα» (δηλ. εκτελούνται με βάση το σχήμα “cobegin Δ1; Δ2; coend”). Θεωρήστε ότι κάθε εντολή εκχώρησης τιμής εκτελείται ατομικά - αδιαίρετα και ότι κάθε εντολή if εκτελείται σε 2 βήματα, ως εξής: βήμα 1: πρώτα ελέγχεται η συνθήκη της if, και βήμα 2: εάν η συνθήκη της if είναι αληθής, μετά εκτελείται η εντολή print. Οι διεργασίες χρησιμοποιούν δύο κοινά διαμοιραζόμενες ακέραιες μεταβλητές (τις x και y).

```
shared int x, y; /*αρχικά έχουν οποιαδήποτε τιμή, δεν παίζει ρόλο*/
```

Διεργασία Δ1	**Διεργασία Δ2**

`x = 1;`
`y = 2;`
`if (x == y)`
`  print “X”;`
	`x = 2;`
	`y = 2;`
	`if (x == y)`
	`  print “Y”;`

[i] Να καθορίσετε όλα τα τελικά αποτελέσματα που είναι δυνατό να εμφανιστούν μετά το πέρας της παράλληλης εκτέλεσης των διεργασιών Δ1 και Δ2. **[5]**

Σημείωση: Δεν απαιτείται να προσδιορίσετε όλα τα δυνατά σενάρια σειράς εκτέλεσης-αναμείξεων (interleavings) των εντολών των διεργασιών. Θα σας βοηθήσει να προσδιορίσετε ένα (1) σενάριο εκτέλεσης για κάθε τελικό αποτέλεσμα που είναι δυνατό να εμφανιστεί.

[ii] Να χρησιμοποιήσετε σημαφόρους (αρχικοποιώντας τους κατάλληλα), και εντολές signal (ή up ή V) και wait (ή down ή P) στους σημαφόρους αυτούς, ώστε να εξασφαλίζεται ότι μετά το τέλος της παράλληλης εκτέλεσης των διεργασιών Δ1 και Δ2 εμφανίζεται πάντοτε το αποτέλεσμα “XY”. **[5]**

Ερώτημα Γ [10]

Θεωρήστε τον ακόλουθο κώδικα για δύο διεργασίες Δ0 και Δ1 που εκτελούνται «παράλληλα» (δηλ. εκτελούνται με βάση το σχήμα “cobegin Δ0; Δ1; coend”).

```
shared boolean flag[2]; /* αρχικά flag[0]=flag[1]=FALSE*/
shared turn[2];         /* αρχικά turn[0]=0 και turn[1]=0*/
```

Διεργασία Δ0

```
flag[0]=TRUE;
turn[0]=(turn[1]+0)mod2;
repeat noop
until (flag[1]==FALSE)OR(turn[0]<>(turn[1]+0)mod2);
<ΚΡΙΣΙΜΟ ΤΜΗΜΑ>
flag[0]= FALSE;
```

Διεργασία Δ1

```
flag[1]=TRUE;
turn[1]=(turn[0]+1)mod2;
repeat noop
until (flag[0]==FALSE)OR(turn[1]<>(turn[0]+1)mod2);
```

```
<ΚΡΙΣΙΜΟ ΤΜΗΜΑ>
flag[1]= FALSE;
```

Σημείωση: Η εντολή noop στον ανωτέρω κώδικα σημαίνει ότι δεν εκτελείται τίποτα (no- operation).

Να εξετάσετε, δίνοντας την απαραίτητη τεκμηρίωση, εάν ο παραπάνω κώδικας εξασφαλίζει (ή όχι) αμοιβαίο αποκλεισμό για τις διεργασίες Δ0 και Δ1, δηλαδή εάν υπάρχει ή δεν υπάρχει περίπτωση οι δύο διεργασίες να εκτελούν ταυτόχρονα το τμήμα εντολών <ΚΡΙΣΙΜΟ ΤΜΗΜΑ> του κώδικά τους.

ΘΕΜΑ 3 [25 μονάδες]

Ερώτημα Α [10]

Τρεις φοιτητές μελετούν από κοινού στη βιβλιοθήκη του πανεπιστημίου τους για να εκπονήσουν μια εργασία. Οι φοιτητές προσομοιώνονται με τρεις αντίστοιχες διεργασίες που παρουσιάζονται στον παρακάτω κώδικα (“Student_1”, “ Student_2” και “ Student_3”), οι οποίες όταν εκτελούνται «παράλληλα» (δηλ. όταν εκτελούνται με βάση το σχήμα “cobegin Student_1; Student_2; Student_3; coend”) θα πρέπει να εκτελούν το ακόλουθο σενάριο συγχρονισμού.

Κάθε φοιτητής από τους τρεις αναζητά στα ράφια της βιβλιοθήκης ένα σχετικό βιβλίο με την εργασία (εκτελεί τη ρουτίνα “Search_Book()” στον κώδικα της αντίστοιχης διεργασίας). Εάν οι άλλοι δύο φοιτητές δεν έχουν βρει ένα σχετικό βιβλίο θα πρέπει να τους περιμένει (εκτελώντας τις εντολές στο σημείο “???” του κώδικα της αντίστοιχης διεργασίας). Ο φοιτητής κάθεται στο τραπέζι του αναγνωστηρίου της βιβλιοθήκης και μελετά την εργασία μαζί με τους άλλους δύο φοιτητές (εκτελεί τη ρουτίνα “Study_Project()” στον κώδικα της αντίστοιχης διεργασίας).

Διεργασία *Student_1*	**Διεργασία *Student_2***	**Διεργασία *Student_3***

`Search_Book();`
`???`
`Study_Project();`
	`Search_Book ();`
	`???`
	`Study_Project();`
		`Search_Book ();`
		`???`
		`Study_Project();`

Καλείστε να συμπληρώσετε τον κώδικα των τριών διεργασιών στα σημεία με “???” με εντολές signal (ή up ή V) και wait (ή down ή P) σε σημαφόρους (που πρέπει να αρχικοποιήσετε κατάλληλα), ώστε να εξασφαλίζεται το πιο πάνω σενάριο συγχρονισμού.

Ερώτημα Β [15]

Καλείστε ως προγραμματιστής να υλοποιήσετε τη Βάση Δεδομένων (ΒΔ) του πληροφοριακού συστήματος της βιβλιοθήκης του πανεπιστημίου. Λάβετε υπόψη ότι στη ΒΔ πρέπει να έχουν ταυτόχρονη πρόσβαση τρία είδη διεργασιών:

α) Διεργασίες που υλοποιούν αναζήτηση βιβλίων στη ΒΔ. Μια αναζήτηση βιβλίου μπορεί να εκτελείται ταυτόχρονα με άλλες αναζητήσεις βιβλίων.

β) Διεργασίες που υλοποιούν εισαγωγή βιβλίων στη ΒΔ. Μια εισαγωγή βιβλίου δεν πρέπει να εκτελείται ταυτόχρονα με άλλες εισαγωγές βιβλίων. Ωστόσο μια εισαγωγή βιβλίου μπορεί να εκτελείται ταυτόχρονα με αναζητήσεις βιβλίων.

γ) Διεργασίες που υλοποιούν διαγραφή βιβλίων. Μια διαγραφή βιβλίου δεν πρέπει να εκτελείται ταυτόχρονα με άλλες διαγραφές βιβλίων καθώς επίσης δεν πρέπει να εκτελείται ταυτόχρονα με αναζητήσεις βιβλίων και με εισαγωγές βιβλίων.

Με βάση τις παραπάνω απαιτήσεις συγχρονισμού, ένας προγραμματιστής συνάδελφός σας έχει ήδη υλοποιήσει «παράλληλο» κώδικα με χρήση σημαφόρων για τις διεργασίες της αναζήτησης και της διαγραφής βιβλίων. Συγκεκριμένα, υλοποίησε τις διεργασίες “Searching” και “Deletion” που όταν εκτελούνται «παράλληλα» (δηλ. όταν εκτελούνται με βάση το σχήμα “cobegin Searching; Deletion; coend”) συγχρονίζουν την εκτέλεση αντίστοιχων έτοιμων (διαθέσιμων) ρουτινών (“Search_Book()” και “Delete_Book()” προκειμένου να ικανοποιούνται οι παραπάνω απαιτήσεις. Ο κώδικας των διεργασιών “Searching” και “Deletion” παρουσιάζεται ακολούθως.

```
binary semaphore mutex = 1;
binary semaphore library = 1;
int count = 0;
```

Searching

```
while (TRUE) {
```

Deletion

```
while (TRUE) {
```

```
    down(mutex);                          down(library);
    count = count + 1;                    Delete_Book();
    if (count == 1) down(library);        up(library);
    up(mutex);                           }
    Search_Book();
    down(mutex);
    count = count - 1;
    if (count == 0) up(library);
    up(mutex);
  }
```

Θα πρέπει να μελετήσετε τον παραπάνω κώδικα και στη συνέχεια να υλοποιήσετε και τον κώδικα για τις διεργασίες “Insertion”, που πραγματοποιούν εισαγωγή βιβλίων στη ΒΔ. Μπορείτε να χρησιμοποιήσετε επιπρόσθετους σημαφόρους και επιπρόσθετες μεταβλητές αλλά δεν πρέπει να τροποποιήσετε τον παραπάνω κώδικα για τις διεργασίες “Searching” και “Deletion”. Ο κώδικας των διεργασιών “Insertion” που θα υλοποιήσετε θα χρησιμοποιεί τη ρουτίνα “Insert_Book()”, την οποία θεωρήστε ότι είναι διαθέσιμη. Οι διεργασίες “Insertion” (τον κώδικα των οποίων θα υλοποιήσετε) όταν εκτελούνται «παράλληλα» με τις διεργασίες “Searching” και “Deletion” (δηλ. όταν εκτελούνται με βάση το σχήμα “cobegin Searching; Deletion; Insertion; coend”) θα πρέπει να ικανοποιούν τις απαιτήσεις συγχρονισμού που αναφέρθηκαν αρχικά στην εκφώνηση.

ΘΕΜΑ 4 [10 μονάδες]

Πέντε διεργασίες καταφθάνουν σε ένα υπολογιστικό σύστημα σύμφωνα με τα δεδομένα του ακόλουθου πίνακα:

Όνομα Διεργασίας	Χρονική Στιγμή Άφιξης	Απαιτήσεις Χρόνου Εκτέλεσης	Προτεραιότητα
Α	0	7	2
Β	3	11	3
Γ	9	5	1
Δ	12	14	5
Ε	14	4	4

Να σχεδιάσετε τα διαγράμματα εκτέλεσης των διεργασιών (Gantt) και να υπολογίσετε τους αντίστοιχους μέσους χρόνους διεκπεραίωσης (ΜΧΔ) και μέσους χρόνους αναμονής (ΜΧΑ), για καθέναν από τους εξής αλγόριθμους χρονοδρομολόγησης:

- FCFS (First Come First Served). [2 μονάδες]
- SJF (Shortest Job First). [2 μονάδες]
- SRTF (Shortest Remaining Time First). [2 μονάδες]
- Priority Scheduling – μη προεκχωρητικός (non-preemptive priority). [2 μονάδες]
- RR (Round Robin) με κβάντο χρόνου 4 χρονικές μονάδες. [2 μονάδες]

Παραδοχές:

1. Ο χρόνος εναλλαγής θέματος (context switch) είναι αμελητέος.
2. Μεγαλύτερη τιμή προτεραιότητας σημαίνει μεγαλύτερη προτεραιότητα.
3. Αν τη χρονική στιγμή άφιξης μιας νέας διεργασίας διακόπτεται η εκτέλεση μιας παλιότερης (πχ. επειδή τελειώνει το κβάντο χρόνου της), τότε η νέα διεργασία εισέρχεται στην ουρά των έτοιμων διεργασιών πριν από την παλιά που διακόπηκε.

Καλή Επιτυχία!!!